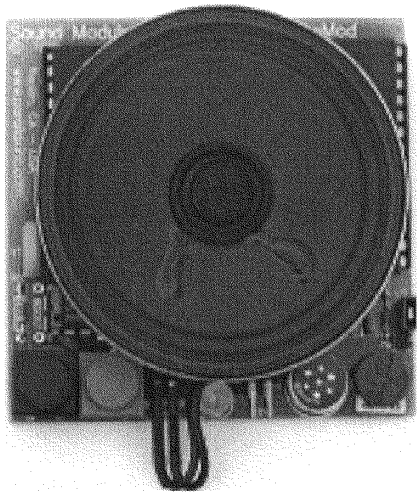
Sound Module AppMod



General Description

There is little doubt that the addition of sound or music can greatly enhance any project. Historically, this has been a difficult task to accomplish. The ICs required to do so are expensive and difficult to work with, especially for beginners. The advent of ISD's family of 'Direct Analog STorage' (DAST) chips has brought the whole process one step closer to the public. And now the Sound Module makes it so easy that everyone can implement audio record and playback capability into their projects.

Features

- **Records/plays up to 60 seconds of sound, as many messages or one big message.**
- **Simple serial interface 'auto-bauds' any standard baudrate from 2400 to 38K4.**
- **Two pushbuttons are provided for local operation.**
- **Input is selectable: microphone or line-in.**
- **1W Audio amplifier and volume control and line-out connection.**
- **Connections provided for optional 'BOOST' capacitor to make it really loud.**
- **New, easy to use write enable/disable feature protects against accidental recording.**

How it Works

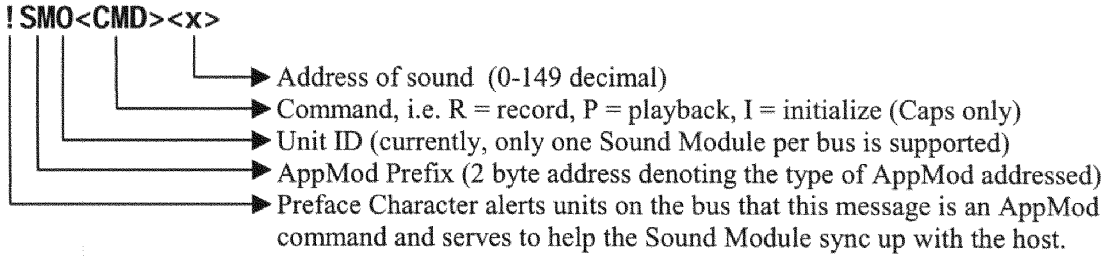
Understanding a little bit more about how the ISD chips works will help you produce better recordings and allow you to use the Sound Module to its fullest potential.

The ISD chip operates at 8 kHz. This means that when the chip is recording, every 122 uSec, (0.000122 Sec), the ISD chip samples and stores the analog input value. Recording is initialized by a low signal on the CE (chip enable) pin on the ISD chip while the P/R (Play/Record selector) pin is held low. When recording ends (CE goes high), the ISD chip automatically appends an EOM (end of message) tag. Playback is initiated by a low going pulse on the CE pin while the P/R pin is high. When playing back, the ISD chip reads the cells from its memory and updates the analog output every 122 uSec until the EOM tag is encountered, then playback is terminated. At the beginning of both playback and record, the address bus is read to determine the location in memory to start recording or playing back. Fear not, the firmware chip on board controls the address bus and handles all of these details for you.

* ISD and DAST are registered trademarks of Information Storage Devices, Inc.

Serial Protocol

The host transmits commands serially to the Sound Module. The Sound Module will not reply unless it has been configured to output an EOM (end of message) pulse. The EOM pulse will be discussed later in this document. The command syntax is structured to facilitate the allowed commands and prevent bus contention in the event that more than one AppMod is on the same serial line. The Sound Module uses P4 for the serial interface.


!SMO<CMD><x>
Address of sound (0-149 decimal)
Command, i.e. R = record, P = playback, I = initialize (Caps only)
Unit ID (currently, only one Sound Module per bus is supported)
AppMod Prefix (2 byte address denoting the type of AppMod addressed)
Preface Character alerts units on the bus that this message is an AppMod command and serves to help the Sound Module sync up with the host.

Commands

C Configure the sound module as specified by x.

serout 4,188+$8000,["!SM0C",1] **'EOM pulse enabled**

This command will configure the Sound Module to pulse the serial data line low for 128 mSec at the end of a message that is played back. Some applications require that the host unit (usually a BASIC Stamp) knows when each message has completed playback. While in this mode, the Sound Module will not listen for serial commands until after both the playback and the EOM pulse have completed.

serout 4,188+$8000,["!SM0C",0] **'EOM pulse disabled**

This command will revert the Sound Module to its default configuration: no EOM pulse.

R Record starting at the address specified by the **x** parameter (in this case **x** = 0).

serout 4,188+$8000,["!SM0R",0]

This command will set the Sound Module's internal address pointer to point to message slot 0 (the very beginning of the sound memory), and start recording. It is up to the user to stop the recording process sometime before the end of sound memory is reached.

I Initialize the Sound Module and set the internal address pointer to point to **x**.

serout 4,188+$8000,["!SM0I",0]

This command will set the Sound Modules internal address pointer to point to message slot 0, which is the beginning of the sound memory. The Sound Module will default to playback mode, but will play no message. The sound chip will be reset, but will retain any sound recorded. This command may be used to stop a record session or a playback session.

M Manually set the address pointer to x.

serout 4,188+$8000,["!SM0M",0]

This command will set the Sound Modules internal address pointer to point to message slot 0, the beginning of memory. No message will play. This command is handy when recording or listening to messages using the on-board pushbuttons. This command neither starts nor stops a playback or record session.

P Playback the sound at the address specified by **x**.

serout 4,188+$8000,["!SM0P",75]

This command will set the Sound Modules internal address pointer to point to message slot 75. 75 * 0.4 Seconds/slot = 30.0 seconds into the sound memory or ½ way through the memory. The message will play until the Sound Module receives another command, or when the end of that particular message is reached.

E End the current playback now.

serout 4,188+$8000,["!SM0E"] **'Stop the playing or recording.**

This command will cause the Sound Module to stop playing or recording a message. Please note that there is no parameter needed for this command.

N Enable recording.

serout 4,188+$8000,["!SM0N"] **'Enable the sound module record mode.**

This command will allow the Sound Module to record messages. Please note that there is no parameter needed for this command. The LED will be left as either RED (record mode) or GREEN (playback mode) after receiving this message.

D Disable recording.

serout 4,188+$8000,["!SM0D"] **'Disable recording.**

This command will prevent the Sound Module from recording messages. Please note that there is no parameter needed for this command. The LED will be toggle between RED and GREEN after receiving this command to indicate the record-disabled state. Playback will function normally.

Pushbutton Mode

Local operation is possible using the two pushbuttons located on the Sound Module. The Black pushbutton switches the mode. The mode is indicated by the bi-color LED. When the LED is green, the Sound Module is in playback mode. When the LED is RED, the Sound Module is in RECORD mode. The Black pushbutton causes the action to occur. When the LED is toggling between RED and GREEN, the unit is in protected mode and recording is not possible. The following will guide you through recording and playing a message.

First you must enable recording by sending a serial message to the Sound Module. Here is the BS2 code that will accomplish this:

```
serout 4, 84+$8000, ["!SM0N"]
```

Recording from the microphone is accomplished by first selecting the microphone as the source of recording by positioning the shunt on X2 high, thereby shorting pins 1 & 2. Pin 1 is closest to the PCB text "X2". Next, you need to push the black button until the LED is RED. The last step is to press the blue button to record. You must hold the blue button as long as you wish to record. Release the blue button when you are finished recording.

Recording from the line input is accomplished by first selecting the line input as the source of recording by positioning the shunt on X2 low, thereby shorting pins 2 & 3. Pin 3 is farthest from the PCB text "X2". Connect the line level source to the +In- pins observing polarity. The amplitude of the incoming signal should be close to 90mVp-p. When you are ready to record, you need to push the black button until the LED is RED. The last step is to push and hold the blue button for the duration of the record time.

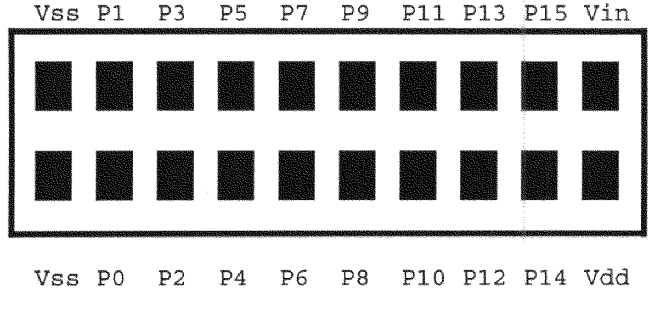
Playback is accomplished by first pushing the black button until the LED is green. Then momentarily push the blue button once. The current message will play until its end. To play a different message you must first use the 'M' command to set the address pointer manually.

Electrical Specifications

The Sound Module AppMod requires 6 – 12 Volts DC and will draw 29mA (idle) to 90mA (full blast). When using an external speaker (8 Ohms only), it is best that the supplied voltage be 12 Vdc to ensure the best sound quality, although it will work properly at 6 Vdc. .

As with all AppMods, the Sound Module AppMod has a special stack-through 2x10 header. This header allows you to connect the Sound Module to either the GrowBot, the BOE-Bot, the Stamp Activity Board, the Super Carrier Board, and of course, other AppMods. We've provided the pin out of the AppMod header below, as viewed from the top.

If you will not be connecting this AppMod to a board with the mating stack-through header, all you need to do is: connect Vss to ground, connect Vin to + 6-12 Vdc supply, and connect P4 to the serial host. **Please note: P4 can tolerate a voltage swing from 0 to 5 Vdc only**.



Although we have provided a stack-through header so that AppMods may be piled on top of one another, the Sound Module should be the top unit since the speaker covers most of the top of the AppMod header.

Programming Notes

1. Throughout the examples mentioned herein, I/O Pin 4 has been referenced as the serial pin for all serial communications. This is because, with respect to the AppMod header included on various products such as the GrowBot, the Board of Education, the BOE-Bot, and the SuperCarrier Board, I/O Pin 4 has been hardwired to serve this function.

2. The examples mentioned herein assume the use of the BS2-IC. Different commands or parameters may be required of you are using something other than the BS2-IC. Please refer to the AppMod code section on the Parallax CD for code examples for the other stamps.

3. When recording, care must be used to prevent recording beyond the end of sound memory. Should this occur, you must place an EOM marker at the end of you message. To do this, write this code and execute it once. After that, the message should play normally.

```
serout 4,188+$8000,["!SM0R",149]
pause 100
serout 4,188+$8000,["!SM0I",0]
```

4. Please note that the 0 in the SM0P command is a numeric (zero) and not an alphabet character. Care must be taken since the two characters look alike in this font. The zero has no function in this AppMod. Others AppMods may use this character location to specify a module number such that multiple AppMods of the same type may share an I/O line.

5. When using the line input for recording, please limit the input voltage to 90 mVpp. Higher voltages will cause distorted recordings and may damage the Sound Module.

6. If you need to additional amplification, you may add a 10 uF 16V electrolytic capacitor across the pads called "BOOST". The negative lead of the cap should be soldered to the pad closest to the 'B' of "BOOST". The gain of the amplifier is increased tenfold. The maximum sound output rating jumps from 26dB to 46dB. The on-board speaker should NOT be used when the BOOST capacitor is installed.

7. To record multiple messages, you must first send a serial command (the 'M' command) to set the address pointer in the sound chip (write the program such that it sends this only once). Then you may record the message manually. Since the address pointer is still set to the beginning of the message, you may switch modes and listen to what you have just recorded. If it is satisfactory to you, you should then change the host program to send an 'M' command with the next address to record to (again, send this command only once). Note: another way to accomplish this is to record all of the sounds as one continuous recording. When playing back the message(s), send a command to start playing at a particular location, wait for a period of time, then send an 'E' (end) command to stop playing the message, or send another 'P' command with a new address to start playing. The effect is to playback the message in specific segments in any order. The talking clock application exemplifies this technique.

8. If the sound module seems to ignore every message you send it, it is probably due to a 'not-so-obvious' syntax error: serout 4,84+$8000,["!SMOP",0]. Can you spot the error? The header SMOP should be SM0P (ess em zero pea) not SMOP (ess em oh pea). Depending on what font you use, 0s and Os can look exactly alike. See note 4.